

kernKill

Loadable Linux kernel module

@vk_khr_ptr

USB Kill

Not USB Killer

```

root@localhost: ~/Desktop/usbskill-master/usbskill

File Edit View Search Terminal Help

root@localhost:~/Desktop/usbskill-master# ./setup.py install
running install
running build
running build_py
running build_scripts
creating build
creating build/scripts-2.7
copying and adjusting /root/Desktop/usbskill-master/install/usbskill -> build/scripts-2.7
changing mode of build/scripts-2.7/usbskill from 644 to 755
running install_lib
warning: install_lib: 'build/lib.linux-i686-2.7' does not exist -- no Python modules to install

running install_scripts
copying build/scripts-2.7/usbskill -> /usr/local/bin
changing mode of /usr/local/bin/usbskill to 755
running install_data
running install_egg_info
Removing /usr/local/lib/python2.7/dist-packages/usbskill-1.0_rc.4.egg-info
Writing /usr/local/lib/python2.7/dist-packages/usbskill-1.0_rc.4.egg-info
root@localhost:~/Desktop/usbskill-master# ls
build install README.md Resources setup.py usbskill
root@localhost:~/Desktop/usbskill-master# cd usbskill
root@localhost:~/Desktop/usbskill-master/usbskill# ls
__init__.py usbskill.py
root@localhost:~/Desktop/usbskill-master/usbskill# chmod +x usbskill.py
root@localhost:~/Desktop/usbskill-master/usbskill# ls
__init__.py usbskill.py
root@localhost:~/Desktop/usbskill-master/usbskill# ./usbskill.py

  usbskill

```

Target Audience

- Activists;
- Hacktivists;
- Journalists;
- Politics;
- Dissidents.

Why Linux

Windows/Mac → GNU/Linux

- Open Source;
- Provable Security;
- Independent from Tech Corporations.

Why Kernel-Space?

User-space:

- User-space service can be disabled remotely by any kind of software;
- Easy detectable in systems.

Kernel-space:

- Can't be unloaded by unprivileged users;
- Can masquerade as other drivers and modules.

Security pre-requirements

- GNU/Linux system (with modern compatible hardware);
- Software encrypted SSD/HDD via dm-crypt;
- Encrypted, updated and locked UEFI;
- Encrypted, updated and locked BootLoader.

Best class solution: most ThinkPad's

Main Principles

Module architecture

- The module is registered in the kernel as a USB subsystem driver;
 - `.probe = etx_usb_probe()`,
 - `.disconnect = etx_usb_disconnect()`,
- At each probe and disconnect:
 - Check `id→idVendor == USB_VENDOR_ID && id→idProduct == USB_PRODUCT_ID`

`kill_proc_info()`,
`kernel_power_off()`,
`etc.`

Anti-Forensic Perspective

You're in trouble, what's next?

If Security pre-requirements OK that's enough;

Possible threats:

- Cold boot attacks (get encryption keys from memory) – really rare;

Hackathon Progress

- MVP:
 - USB sub-system registration;
 - Probe and Disconnect detection;
 - Device verification;
 - Kernel-shutdown and proc-kill.

Plans

- Debug;
- Verification via S/N;
- Experiments with veracrypt containers;
- Create useful configuration tool.

thx

lazy_static!



@VK_KHR_PTR



github